



AMIT ACADEMY

for Computer Education

SUBJECT – PROGRAMMING WITH PYTHON

Faculty Name – Ankit Rami

Unit-4

FILE HANDLING IN PYTHON

AMIT ACADEMY FOR COMPUTER EDUCATION

Nr Vardayini Mataji Temple, Rupal, Gandhinagar-382630

Email – amitacademy1117@gmail.com

Mobile No – 8460467193

YouTube Link –

<https://www.youtube.com/@ankitramijoinar>

Instagram Link –

<https://www.instagram.com/amitacademy17/>

Facebook Link –

<https://www.facebook.com/aramitacademy/>

Introduction of File handling in python

- ✓ File handling is an integral part of programming. File handling in Python is simplified with built-in methods, which include creating, opening, and closing files.
- ✓ While files are open, Python additionally allows performing various file operations, such as reading, writing, and appending information.
- ✓ Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters and they form a text file.
- ✓ Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with the reading and writing files.

Opening Files in Python (Working of `open()` function)

- ✓ Before performing any operation on the file like reading or writing, first, we have to open that file. For this, we should use Python's inbuilt function `open()` but at the time of opening, we have to specify the mode, which represents the purpose of the opening file.
- ✓ The `open()` Python method is the primary file handling function.
- ✓ Syntax - `file_object = open('file_name', 'mode')`

PROGRAMMING WITH PYTHON

- ✓ The open() function takes two elementary parameters for file handling:
 1. The file_name includes the file extension and assumes the file is in the current working directory. If the file location is elsewhere, provide the absolute or relative path.
 2. The mode is an optional parameter that defines the file opening method. The table below outlines the different possible options:

Mode	Description
'r'	Reads from a file and returns an error if the file does not exist (default).
'w'	Writes to a file and creates the file if it does not exist or overwrites an existing file.
'x'	Exclusive creation that fails if the file already exists.
'a'	Appends to a file and creates the file if it does not exist or overwrites an existing file.
'b'	Binary mode. Use this mode for non-textual files, such as images.
't'	Text mode. Use only for textual files (default).
'+'	Activates read and write methods.

- ✓ The mode must have exactly one create(x)/read(r)/write(w)/append(a) method, at most one +. Omitting the mode defaults to 'rt' for reading text files.

PROGRAMMING WITH PYTHON

✓ Below is a table describing how each of the modes behave when invoked.

Behavior	Modes
Read	<code>r, r+, w+, a+, x+</code>
Write	<code>r+, w, w+, a, a+, x+</code>
Create	<code>w, w+, a, a+, x, x+</code>
Pointer Position Start	<code>r, r+, w, w+, x, x+</code>
Pointer Position End	<code>a, a+</code>
Truncate (clear contents)	<code>w, w+</code>
Must Exist	<code>r, r+</code>
Must Not Exist	<code>x, x+</code>





Python File Methods

Method	Description
<code>close()</code>	Closes the file
<code>detach()</code>	Returns the separated raw stream from the buffer
<code>fileno()</code>	Returns a number that represents the stream, from the operating system's perspective
<code>flush()</code>	Flushes the internal buffer
<code>isatty()</code>	Returns whether the file stream is interactive or not
<code>read()</code>	Returns the file content
<code>readable()</code>	Returns whether the file stream can be read or not
<code>readline()</code>	Returns one line from the file
<code>readlines()</code>	Returns a list of lines from the file
<code>seek()</code>	Change the file position
<code>seekable()</code>	Returns whether the file allows us to change the file position
<code>tell()</code>	Returns the current file position
<code>truncate()</code>	Resizes the file to a specified size
<code>writable()</code>	Returns whether the file can be written to or not
<code>write()</code>	Writes the specified string to the file
<code>writelines()</code>	Writes a list of strings to the file

Reading Text File using Python

- ✓ After importing a file into an object, Python offers numerous methods to read the contents.
- ✓ Use the `read()` method on the file object and print the result.

Ex-

```
f = open("file.txt")  
print(f.read())
```

Output – Ankit Rami

- ✓ **Read Parts of the File**
- ✓ Provide a number to the **`read()`** function to read only the specified number of characters:

Ex-

```
f = open("file.txt")  
print(f.read(5))
```

Output – Ankit

- ✓ The output prints the first five characters in the file.
- ✓ Alternatively, use the **`readline()`** method to print only the first line of the file:

Ex-

```
f = open("file.txt")  
print(f.readline(5))
```

Output – Ankit Rami

- ✓ Add an integer to the `readline()` function to print the specified number of characters without exceeding the first line.

- ✓ Using for loop read text file

Ex-

```
f = open("file.txt")
```

```
for line in f:
```

```
    print(line)
```

Output - Ankit Rami

- ✓ use the readlines() method on the file object.

Ex-

```
f = open("file.txt")
```

```
print(f.readlines())
```

Output - ['Ankit Rami\n', 'Deep Patel']

- ✓ The function returns the list of lines from the file stream.

- ✓ Add an integer to the readlines() function to control the number of lines.

- ✓ A file remains open until invoking the close() function.

It's good practice to close files no longer in use to avoid unpredictable file behavior and corrupted files.

- ✓ To close a file, run the close() method on the file object.

Ex-

```
f = open("file.txt")
```

```
print(f.readlines())
```

```
f.close()
```

Output - Ankit Rami

Writing Data in File using Python

- ✓ Python write data in file and save the data contained in a text file.
- ✓ The write() or writelines() method helps Python write data in files. Python can be written in a file in various ways.
- ✓ **Method 1: Write in a Text file In Python using write()**
- ✓ The file is opened with the open() method in w+ mode within the with block, the w+ argument will create a new text file in write mode with the help of write(). The with block ensures that once the entire block is executed the file is closed automatically.

Ex-

```
f = open("file.txt",w)
f.write("My Name is Ankit Rami")
f.close()
```

Output - Data Write in file.txt file

- ✓ **Method 2: write in a Text file In Python using writelines()**
- ✓ The file is opened with the open() method in w mode within the with block, the argument will write text to an existing text file with the help of readlines(). The with block ensures that once the entire block is executed the file is closed automatically.

Ex-

```
L = ["Ankit\n", "Rami\n", "Rupal\n"]
```

```
# writing to file
```

```
file1 = open('file.txt', 'w')
```

```
file1.writelines(L)
```

```
file1.close()
```

Output - Data Write in file.txt file

✓ **Method 3: Write in a Text file In Python using Append()**

✓ "a" - Append - will create a file if the specified file does not exist

✓ "a" - Append - will append to the end of the file

Ex-

```
f = open("file.txt", "a")
```

```
f.write("Now the file has more content!")
```

```
f.close()
```

#open and read the file after the appending:

```
f = open("demofile2.txt", "r")
```

```
print(f.read())
```

Output - Data Write in file.txt file without overriding old data as it is and add new data in file

Reading file in buffer size using Python

✓ buffering – If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer

PROGRAMMING WITH PYTHON

size. If negative, the buffer size is the system default(default behavior). Default buffer size you can check with the following code:

Ex-

```
import io
print(io.DEFAULT_BUFFER_SIZE)
```

Output –

8192

- ✓ If you look at the function definition of open - `open(name[, mode[, buffering]])`, you'll see that it takes 3 arguments in Python 2, third one being buffering. The optional buffering argument specifies the file's desired buffer size: 0 means unbuffered, 1 means line buffered, any other positive value means use a buffer of (approximately) that size (in bytes). A negative buffering means to use the system default, which is usually line buffered for tty devices and fully buffered for other files. If omitted, the system default is used.
- ✓ For example, If you want to open a file with a buffer size of 128 bytes you can open the file like this –
- ✓ **>>> open('my_file', 'r+', 128)**
- ✓ In Python 3, the function definition of open is: `open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)`. buffering is an optional integer used to set the buffering policy. Pass 0 to switch buffering off (only allowed in binary mode), 1 to select line buffering (only usable in text

PROGRAMMING WITH PYTHON

mode), and an integer > 1 to indicate the size in bytes of a fixed-size chunk buffer. When no buffering argument is given, the default buffering policy works as follows

- ✓ Binary files are buffered in fixed-size chunks; the size of the buffer is chosen using a heuristic trying to determine the underlying device's "block size" and falling back on `io.DEFAULT_BUFFER_SIZE`.
- ✓ "Interactive" text files (files for which `isatty()` returns `True`) use line buffering. Other text files use the policy described above for binary files.

✓ Ex-

```
f = open("file.txt","r",5)
print(f.readlines())
```

Output –

```
['Ankit Rami\n', 'Deep Patel']
```

Write Data to Binary File in Python

- ✓ Writing data to binary file is a way to perform write operation in our file. One thing is common everywhere, that is opening a file. Before performing any operation on file, opening and closing of file is mandatory for successful execution of task. There are two possibilities while writing data to a file ie.
- ✓ (Note: .dat extension means binary file)
- ✓ (i). If binary file `abc.dat` already existing and have some data written in the binary file and we are writing data to the same file `abc.dat`. Then in that case, the data written in the file is overridden with the new data.

PROGRAMMING WITH PYTHON

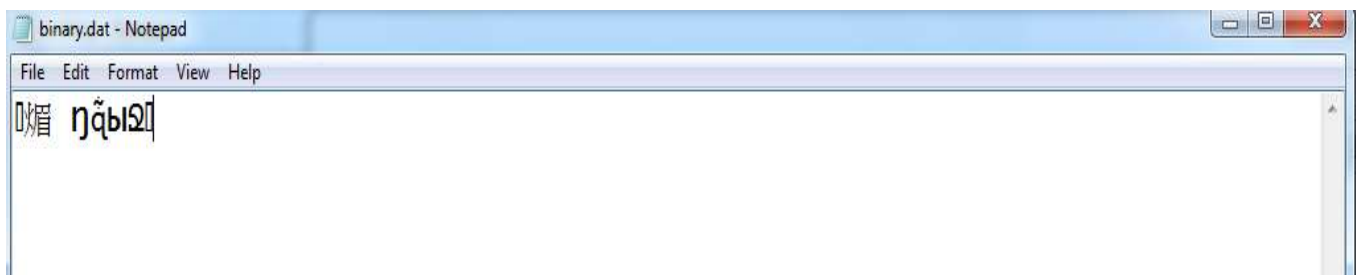
- ✓ (ii). If file is not existing in the system, then it will first create new file with the given name. And then perform the write operation. We can perform write operation on binary file using dump() method available in python pickle library.

✓ Ex-

```
import pickle
def write():
    file = open("binary.dat",'wb')
    x = [1,2,3,4,5] #data we wrote in file
    pickle.dump(x,file)
    file.close()
write()
f1 = open("binary.dat",'rb')
print(f1.read())
```

✓ Output -

b'\x80\x03]q\x00(K\x01K\x02K\x03K\x04K\x05e.'



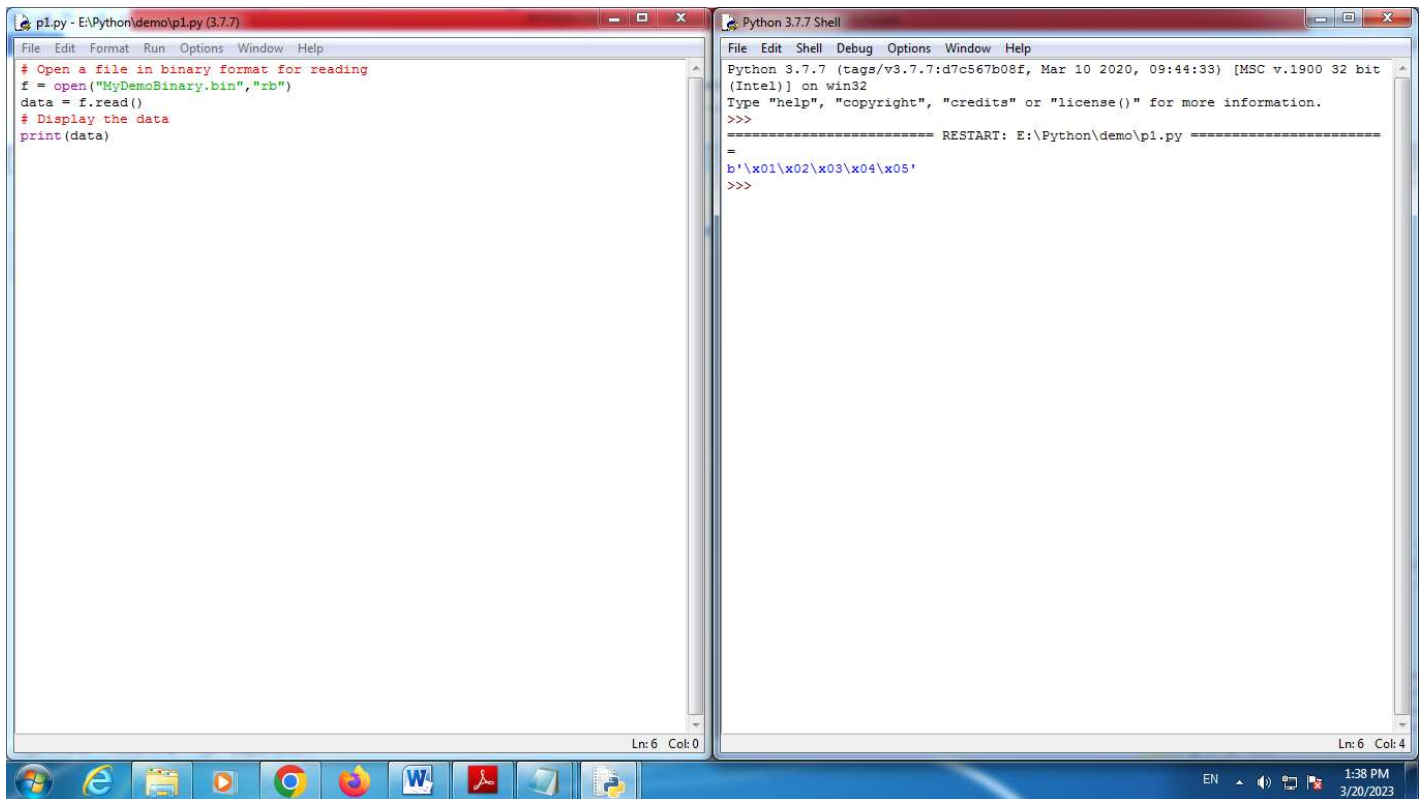
PROGRAMMING WITH PYTHON

- ✓ Method-2 Write to a Binary File
- ✓ The wb mode of the open() method is used to open a file in format form writing.
- ✓ Note – The Binary Files are not huma-readable and the content is unrecognizable
- ✓ Let's see the complete example. Here, the file will get stored in E drive with the name
- ✓ Example
- ✓ MyDemoBinary.bin –

Ex-

```
# Open a file in binary format for writing
f = open("E:\MyDemoBinary.bin","wb")
# Elements to be added to the binary file
a = [100, 200, 300]
# Convert the integer elements to a bytearray
myArr = bytearray(a)
# The byte representation ius now written to the file
f.write(myArr)
f.close()
# Open a file in binary format for reading
fr = open("MyDemoBinary.bin","rb")
data = fr.read()
# Display the data
print(data)
```

Output –

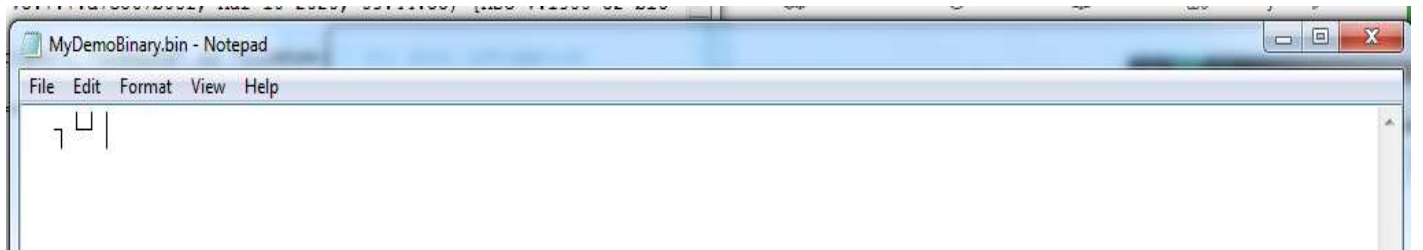


The screenshot shows two windows from a Python IDE. The left window, titled 'p1.py - E:\Python\demo\p1.py (3.7.7)', contains the following Python code:

```
File Edit Format Run Options Window Help
# Open a file in binary format for reading
f = open("MyDemoBinary.bin", "rb")
data = f.read()
# Display the data
print(data)
```

The right window, titled 'Python 3.7.7 Shell', shows the execution output:

```
File Edit Shell Debug Options Window Help
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 09:44:33) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\Python\demo\p1.py =====
=
b'\x01\x02\x03\x04\x05'
>>>
```



The screenshot shows a Notepad window titled 'MyDemoBinary.bin - Notepad'. The window contains the following text:

```
File Edit Format View Help
7 4 |
```

✓ **Renaming Files in Python**

✓ **The rename() Method**

- ✓ The rename() method takes two arguments, the current filename and the new filename.

Ex-

```
import os
```

```
# Rename a file from test1.txt to test2.txt
```

```
os.rename( "test1.txt", "test2.txt" )
```

Output – File Name Change

✓ **The remove() Method**

- ✓ You can use the remove() method to delete files by supplying the name of the file to be deleted as the argument.

Ex-

```
import os
```

```
# Delete file test2.txt
```

```
os.remove("text2.txt")
```

Output – File Deleted

✓ **Copy Files and Directories in Python**

- ✓ Python provides strong support for file handling. We can copy single and multiple files using different methods and the most commonly used one is the shutil.copy() method. The below steps show how to copy a file from one folder to another.

Ex-

```
import shutil
src_path = "E:\Ankit\profit.txt"
dst_path = "E:\account\profit.txt"
shutil.copy(src_path, dst_path)
print('Copied')
```

Output – File Copy

- ✓ **Move Files Or Directories in Python**
- ✓ Python shutil module offers several functions to perform high-level operations on files and collections of files. We can move files using the `shutil.move()` method.
- ✓ Ex –

```
import shutil
# absolute path
src_path = "E:\reports\sales.txt"
dst_path = "E:\account\sales.txt"
shutil.move(src_path, dst_path)
```

Output – File Move

Reference Link

1. <https://www.geeksforgeeks.org/reading-writing-text-files-python/>
2. <https://www.geeksforgeeks.org/file-handling-python/>
3. <https://pynative.com/python/file-handling/>
4. <https://www.tutorialspoint.com/How-do-we-specify-the-buffer-size-when-opening-a-file-in-Python>
5. <https://pylessons.com/Python-3-basics-tutorial-writing-reading-file>



Any Query Contact Us

Faculty Name- Ankit Rami

Email - ankitramiblog@gmail.com

Contact No - +91 8460467193

Website - amit.arinfoaway.com



Subscribe Our YouTube Channel

<https://www.youtube.com/channel/UCWbJh2iQ8w-8nrU0Xpjpw7g>